Lõpmatus, lahenduvus ja mittelahenduvus

Lahenduvus

- Kas aga iga probleemi jaoks on üldse olemas algoritmi, mis seda lahendab?
- Eeldame, et vaatame ainult probleeme, mis on täpselt ja üheselt kirjeldatud ja kus on lahendamiseks olemas piisavalt infot (a la travelling salesman, malemäng jne)
- **Selgub, et iga täpselt formuleeritud probleemi jaoks ei leidugi lahendavat algoritmi!**
- Vähe sellest: kui võtta “juhuslik” probleem, siis tõenäosus, et lahendav algoritm leidub, on lõpmatult väike!

Lahenduvus uurimisvaldkonnana

- Algoritmi- ehk rekursiooniteooria on suur arvutiteaduse uurimisvaldkond.
- Ingliskeelne harilik nimi: Algorithm theory, recursion theory
- Lahenduvus: decidability või computability

- Uuritakse, millistele ülesannetele on algoritme, millistele ei
- Uuritakse, mis ülseande lahendamine taandub teisele ülesandele
- Uuritakse lahendumis, poollahendumist, kreatiivseid hulki jne jne
- Uuritakse lõpmatuse struktuuri, mis on kirjeldamatult keeruline
- Uuritakse lahendumise struktuuri, mis on kirjeldamatult keeruline
- Uuritakse loogikaklasside lahendumise taandumist muudele ülesannetele
- ....

Intuitiivne seletus lahendamatusele

- Saab näidata, et erinevaid probleeme on lõpmatult rohkem, kui erinevaid algoritme.
- Kuna probleeme on lõpmatult rohkem kui algoritme, siis iga probleemi jaoks lihtsalt “ei jätku” lahendavat algoritmi.
- Kuidas seda näidata (plaan):
  - Näitame, et **algoritme on sama palju, kui täisarve** (lihtne)
  - Näitame, et **probleeme on vähemalt sama palju, kui reaalarve** (veidi keerulisem)
  - Näitame, et **reaalarve on lõpmatult rohkem kui täisarve** (Cantori üks teoreeme)

Algoritme on sama palju (või vähem?) kui täisarve

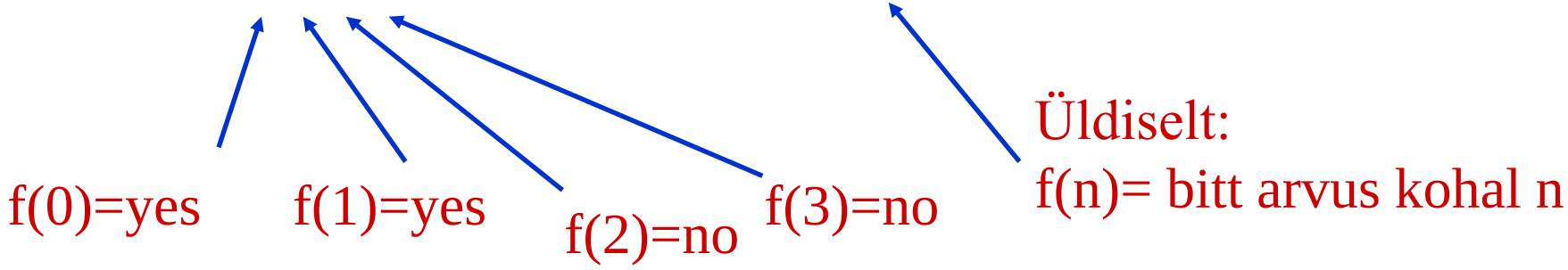
- Iga algoritmi saab kirjutada mistahes programmeerimiskeeles.
- Valime näiteks C keele.
- **Iga C keelne programm on tekstifail, st üks pikk string.**
- String on näiteks “asas asss ddddd”.
- String koosneb järjestikustest baitidest, iga bait vahemikus 0-255
- **Iga string vastab ühele täisarvule:** vaatame stringi kui 256-süsteemis arvu, näiteks:
  - Baidid **0 22 64** annavad arvu **22*256+64**
  - Baidid **64 120 68** annavad arvu **64*256 ruudus + 120*256 + 64**
- NB! Iga string ei ole korrektne C programm. Vastupidi küll.

Probleeme on sama palju kui reaalarve

- **Mis on reaalarv? Arv, kus koma järel võib olla kuitahes palju komakohti.**
- **Näiteks**: 2,23242545344123123 1...
- 2, pi, ¾, ruutjuur kahest on kõik reaalarvud.
- Kahendsüsteemis reaalarvul on iga number kas 0 või 1, näiteks:
  110.11010010110101111010101 01....

- Võtame ühe spetsiifilise klassi probleemidest: “yes-no” probleemid täisarvudel:
  Algoritm võtab sisendiks täisarvu ja peab vastama “yes” või “no”.

- Kui palju selliseid probleeme on?
- Iga kahendsüsteemis reaalarv alla ühe vastab ühele probleemile:
- `0.11001010101010101010101 ...`



Cantori teoreem: sissejuhatus

- **Reaalarvude hulk on suurem (võimsam) kui positiivsete täisarvude hulk.**
- Enne uurime, kuidas on lugu pos/neg täisarvudega ja murdudega.
- Pos/neg täisarve oleks justkui kaks korda rohkem, kui positiivseid täisarve??

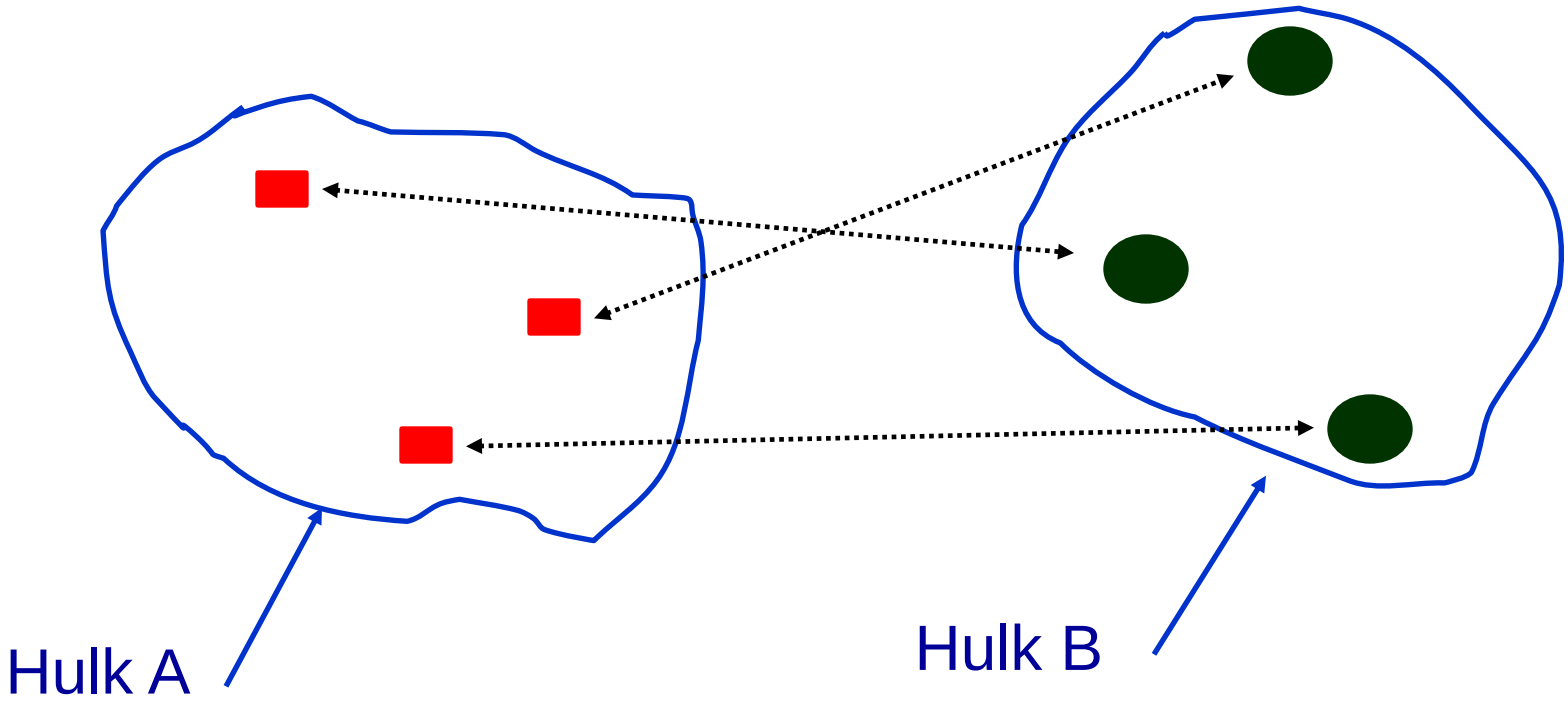
```
... -4   -3   -2   -1    0    1    2    3    4 ...
                        0    1    2    3    4 ...
```

Neg/Pos täisarvud: N

Positiivsed täisarvud: Z

Cantori teoreem: sissejuhatus

- Mida tähendab väide: “hulk A on sama suur (võimas) kui hulk B”?
- Seda, et hulga A kõik elemendid saab seda üksühesesse vastavusse hulga B elementidega:



Igale A elemendile vastab täpselt üks B element ja vastupidi

Cantori teoreem: sissejuhatus

- Kuidas seada üksühesesse vastavusse kahe lõpmatu hulga N ja Z elemendid?

N:	...	-4	-3	-2	-1	0	1	2	3	4 ...
Z:						0	1	2	3	4 ...

- N oleks justkui suurem kui Z?

Cantori teoreem: sissejuhatus

- Kuidas seada üksühesesse vastavusse kahe lõpmatu hulga N ja Z elemendid?

```
N:  ... -4  -3  -2  -1   0   1   2   3   4 ...
Z:                       0   1   2   3   4 ...
```

- Seame vastavusse hoopis niimoodi:

```
N: 0  1  -1  2  -2  3  -3  4  -4  ...
Z: 0  1   2  3   4  5   6  7   8   ...
```

- **Kuna N ja Z saab üksühesesse vastavusse seada, siis on nad sama suured (sama võimsad)!**

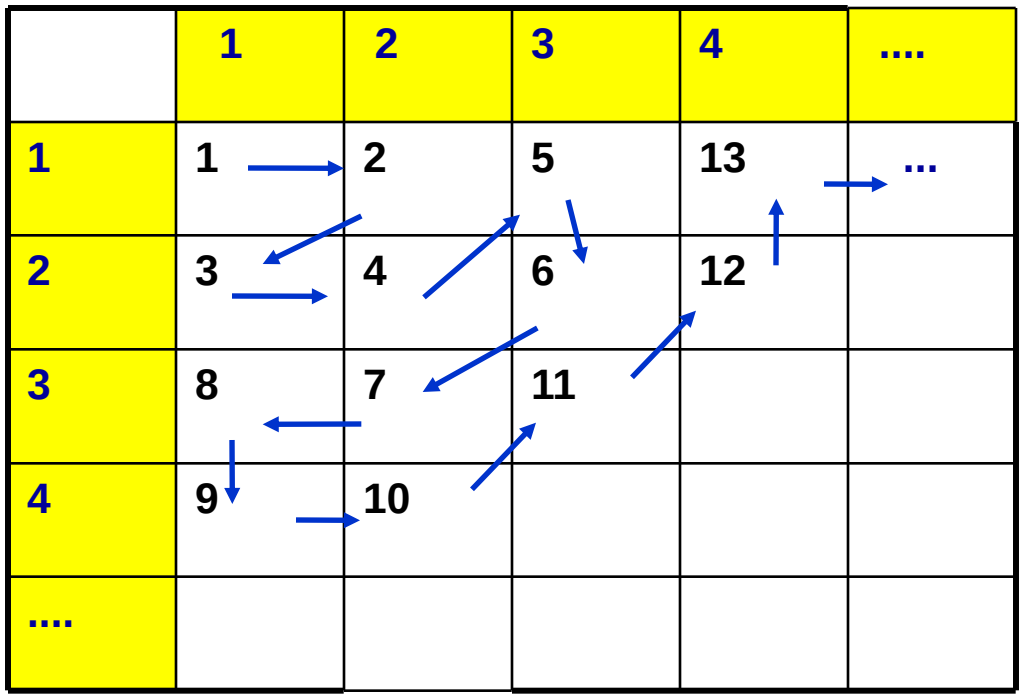
Cantori teoreem: sissejuhatus

- Murdarve oleks justkui lõpmatult rohkem, kui positiivseid täisarve?? Samuti vale!

Tegelikult on murdarvud vs pos täisarvud ükskühesed vastavuses:

Murru kordajad

Jagajad



Cantori teoreem: kõigepealt idee

- Koostame kõigi 1-st väiksemate reaalarvude tabeli:

Arvu kohad pärast koma

arvud						
	1	**2**	**1**	**5**	**6**	**...**
	3	**3**	**1**	**0**	**9**	**...**
	2	**8**	**3**	**5**	**6**	**...**
	3	**5**	**6**	**9**	**0**	**...**
					...	

- Ja nüüd konstrueerime diagonaali (**1 3 3 9 ...**) järgi uue arvu, liites diagonaali arvudele ühe (kui tulemus 10, võtame 0): saame **2 4 4 0 ...**

Cantori teoreem: detailselt läbi tehes 1

Arvu kohad pärast koma

arvud

1	**2**	**1**	**5**	**6**	**...**
3	**3**	**1**	**0**	**9**	**...**
2	**8**	**3**	**5**	**6**	**...**
3	**5**	**6**	**9**	**0**	**...**
				...	

- **Konstrueerime uue arvu esimese numbri:**
  - Esimesel real tulbas 1 oli number 1.
  - Meie võtame 1+1 =2
  - Meie uue arvu esimene number on seega 2:
- **Meie uus arv: 0 . 2 ...**

Cantori teoreem: detailselt läbi tehes 2

Arvu kohad pärast koma

arvud						
1	1	2	1	5	6	...
2	3	3	1	0	9	...
3	2	8	3	5	6	...
4	3	5	6	9	0	...
...					...	

- **Konstrueerime uue arvu teise numbri:**
  - Teisel real tulbas 2 oli number 3.
  - Meie võtame 3+1 =4
  - Meie uue arvu teine number on seega 4:
- **Meie uus arv: 0 . 2 4 ...**

Cantori teoreem: detailselt läbi tehes 3

Arvu kohad pärast koma

arvud

1	**2**	**1**	**5**	**6**	**...**
3	**3**	**1**	**0**	**9**	**...**
2	**8**	**3**	**5**	**6**	**...**
3	**5**	**6**	**9**	**0**	**...**
				...	

- **Konstrueerime uue arvu kolmanda numbri:**
  - Kolmandal real tulbas 3 oli number 3 .
  - Meie võtame 3+1 =4
  - Meie uue arvu kolmas number on seega 4:
- **Meie uus arv: 0 . 2 4 4 ...**

Cantori teoreem: detailselt läbi tehes 4

Arvu kohad pärast koma

1	**2**	**1**	**5**	**6**	**...**
3	**3**	**1**	**0**	**9**	**...**
2	**8**	**3**	**5**	**6**	**...**
3	**5**	**6**	**9**	**0**	**...**
				...	

arvud

- **Konstrueerime uue arvu neljanda numbri:**
  - Neljandal real tulbas 4 oli number 9 .
  - Meie võtame 9+1 = 10, meie võtame siis numbriks 0
  - Meie uue arvu kolmas number on seega 0:
- **Meie uus arv: 0 . 2 4 4 0 ...**

Cantori teoreem: detailselt läbi tehes 5

Arvu kohad pärast koma

arvud						
	1	**2**	**1**	**5**	**6**	**...**
	3	**3**	**1**	**0**	**9**	**...**
	2	**8**	**3**	**5**	**6**	**...**
	3	**5**	**6**	**9**	**0**	**...**
					...	

- Meie uus arv oli: **0 . 2 4 4 0 ...**
- Kas meie uus arv on meie tabelis, st mõni rida tabelis?
- **Igatahes ei saa ta olla esimene rida:**
  - Esimesel real oli esimene arv **1**, meil aga on esimene arv **2**

Cantori teoreem: detailselt läbi tehes 6

Arvu kohad pärast koma

arvud

1	**2**	**1**	**5**	**6**	**...**
3	**3**	**1**	**0**	**9**	**...**
2	**8**	**3**	**5**	**6**	**...**
3	**5**	**6**	**9**	**0**	**...**
				...	

- Meie uus arv oli: **0 . 2 4 4 0 ...**
- Kas meie uus arv on meie tabelis, st mõni rida tabelis?
- **Igatahes ei saa ta olla teine rida:**
  - Teisel real oli teine arv **3**, meil aga on teine arv **4**

Cantori teoreem: detailselt läbi tehes 7

Arvu kohad pärast koma

arvud						
	1	**2**	**1**	**5**	**6**	**...**
	3	**3**	**1**	**0**	**9**	**...**
	2	**8**	**3**	**5**	**6**	**...**
	3	**5**	**6**	**9**	**0**	**...**
					...	

- Meie uus arv oli: **0 . 2 4 4 0 ...**
- Kas meie uus arv on meie tabelis, st mõni rida tabelis?
- **Igatahes ei saa ta olla kolmas rida:**
  - Kolmandal real oli kolmas arv **3**, meil aga on kolmas arv **4**

Cantori teoreem: detailselt läbi tehes 8

Arvu kohad pärast koma

arvud

1	**2**	**1**	**5**	**6**	**...**
3	**3**	**1**	**0**	**9**	**...**
2	**8**	**3**	**5**	**6**	**...**
3	**5**	**6**	**9**	**0**	**...**
				...	

- Meie uus arv oli: **0 . 2 4 4 0 ...**
- Kas meie uus arv on meie tabelis, st mõni rida tabelis?
- **Igatahes ei saa ta olla neljas rida:**
  - Neljandal real oli neljas arv **9**, meil aga on neljas arv **0**

Cantori teoreem: detailselt läbi tehes 9

Arvu kohad pärast koma

arvud						
	1	**2**	**1**	**5**	**6**	**...**
	3	**3**	**1**	**0**	**9**	**...**
	2	**8**	**3**	**5**	**6**	**...**
	3	**5**	**6**	**9**	**0**	**...**
					...	

- Selline asjade käik ei olnud juhus: me ise konstrueerisime oma arvu!
- Üldiselt on (meie enda konstruktsiooni-meetodi järgi) nii:
- **Igatahes ei saa ta olla N-s rida:**
  - N-ndal real oli N-s arv **X**, meil aga on N-s arv **X+1 (kui X= 9, siis 0)**

Cantori teoreem: detailselt läbi tehes 10

- **Kokkuvõttes:**

- Meie arv 0. 2 4 4 0 .... ei saa olla selles tabelis
- Kui meil oleks tabel kuidagi teisiti tehtud (arvud teises järjekorras) siis:
  - kui me jälle teeksime diagonaali järgi oma uue arvu, siis seda arvu ikka tabelis ei ole.
- Meie uus arv kahtlemata on reaalarv.
- Seega ei sisalda ükski 1-st väiksemate reaalarvude tabel kõiki reaalarve!
- Mis see siis tähendab:
  - Kõiki reaalarve ei saagi tabelisse panna
  - Iga tabeli N-s rida vastab täisarvule N
  - **Reaalarvude hulk on suurem (võimsam) kui täisarvude hulk**

Cantori teoreemi edasine jätk

Pane tähele, et:

- Iga kahendsüsteemis reaalarv 0 ja 1 vahel **vastab ühele täisarvude alamhulgale**: N-nda biti positsioon ütleb, kas arv N on seal hulgas või ei.

- **Näiteks:**
  - Paarisarvude hulgale vastab: `10101010101010101010....`
  - Kõigi arvude hulgale vastab: `11111111111111111....`
  - Algarvude hulgale vastab: `01010101000101....`

- Cantori teoreem ütleb üldisemalt, et **mingi hulga H kõigi alamhulkade hulk on suurema võimsusega kui see hulk H.**

- **Tekivad lõpmatud ahelad üha suurema võimsusega hulkadest:**
- Täisarvude hulk N1 → N1 alamhulkade hulk N2 → N2 alamhulkade hulk N3 → jne jne ....

Kontiinumhüpotees (CH)

Cantori oletus aastast 1878. Siiani tõestamata/ümberlükkamata!

- Täisarvude lõpmatuse ja reaalarvude lõpmatuse vahel ei ole teisi lõpmatusi !?!?

- In 1963, Paul Cohen proved that the hypothesis is independent of the axioms of Zermelo–Fraenkel set theory with the axiom of choice (ZFC), a standard axiomatization of set theory, complementing earlier work by Kurt Gödel in 1940. Such independence means that ZFC can be augmented by either CH or its negation ¬CH, in both cases producing a system of axioms that is consistent if and only if ZFC is consistent.

Konkreetne näide mittelahenduvusest: Halting Problem

- Problem:You are an employee of the software company Gigasoft, which is writing a new operating system.
- **Your boss wants you to write a program that will take in a user's program and inputs and decide whether**
  - it will eventually stop, or
  - it will run infinitely in some infinite loop.
- If the program will run infinitely, your program will disallow the program to run and send the user a rude message.

- **Call this problem the Halting Problem.**

Attempts

- **Look for loops such as *while (statement)***
  - If variables in the statements are unchanged e.g.
    - *While (t<1)* with *t* is initialized to *0* but never used in the loop
    - No statements to get out of the loop, such as *goto* or *break*
  - The program will not halt.
- **What about programs like:**

```
TestGolbach
    i=2; c=true;
    while (c==true)
        i=i+2;
        find all primes smaller than i, put in a set S;
        is_sum = false;
        for each pair (p,q) of primes in S
            if (i==p+q) is_sum=true;
        if (is_sum==false) c=false;
```

  - Will TestGolbach halt?

Is there a solution?

- **If you can solve the halting problem, you can solve Golbach's conjecture:**
  - Every even number greater than 2 can be represented as a sum of two primes.
- **Golbach's conjecture is an unsolved problem in mathematics**
- **What does this tell you about the halting problem?**
- **What should you do?**
  - One possibility: try to show that the halting problem is not solvable.

Proof: start

- Assume that it is possible to write a program to solve the Halting Problem.
- Denote this program by **HaltAnswerer(*prog*,*inputs*).**
- **HaltAnswerer(*prog*,*inputs*) will**
  - **return *yes* if *prog* will halt on *inputs* and**
  - ***no* otherwise.**
- A program is just a string of characters
  - E.g. your Java program is just a long string of characters
- An input can also be considered as just a string of characters
- So HaltAnswerer is effectively just working on two strings

Proof part 2:

- We can now write another program **Nasty(prog)** that uses **HaltAnswerer** as a subroutine
- The program **Nasty(prog)** does the following:

  [1] If HaltAnswerer(prog,prog) returns yes,
  Nasty will go into an infinite loop

  [2] If HaltAnswerer(prog,prog) returns no,
  Nasty will halt

- Consider what happens when we run **Nasty(Nasty)**.
- If **Nasty loops infinitely**,
  - HaltAnswerer(Nasty,Nasty) returns no which by [2] above means Nasty will halt.
- If **Nasty halts**,
  - HaltAnswerer(Nasty,Nasty) will return yes which by [1] above means Nasty will loop infinitely.

- Conclusion: Our assumption that it is possible to write a program to solve the Halting problem has resulted in a contradiction.

Diagonalization for halting problem: try it out!!

- **Each program can be represented by a string and each string can be represented by a natural number**
- **Create a table for all programs with a single input where the entry *(i,j)* is**
  - *H* if program *i* halts when program *j* is used as input
  - *NH* if program *i* does not halt when program *j* is used as input
- **Where is *Nasty* in this list?**

Kuulus näide matemaatikast

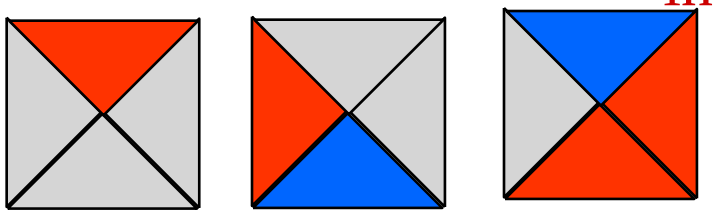
- In 1900, mathematician David Hilbert identified 23 mathematical problems and posed them as a challenge for the coming century.
- **The tenth problem asks for an algorithm to test whether a polynomial has an integral root**
- Apparently, Hilbert assumed that such an algorithm must exist.
- **This problem is unsolvable** (Matijasevic 1970)

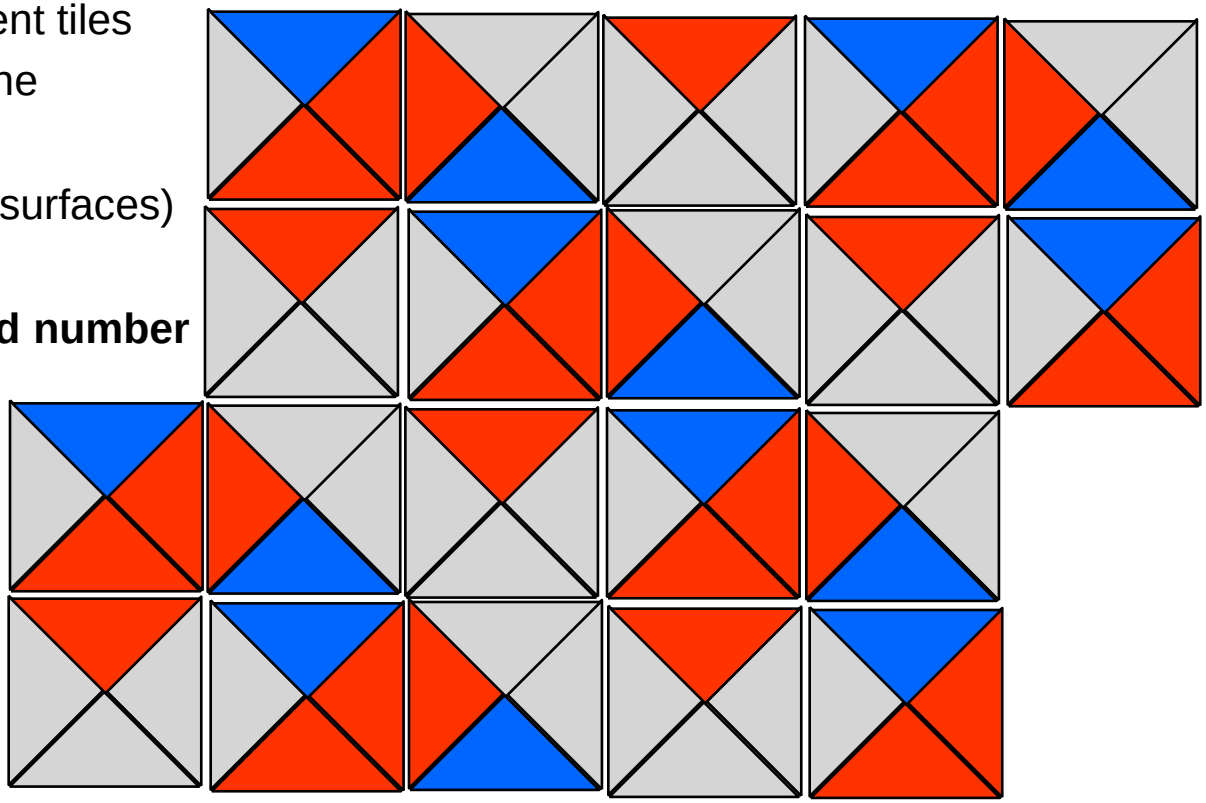
Lihtne näide geomeetriast: “tiling problem”

Assume that you have
a finite set of tiles
(which cannot be rotated)
and you would like
to know if you can tile
an area using those
tiles (adjacent tiles
must have the
same color
on adjoining surfaces)

An **unlimited number of tiles** is available of each kind

Three tiles given in this example

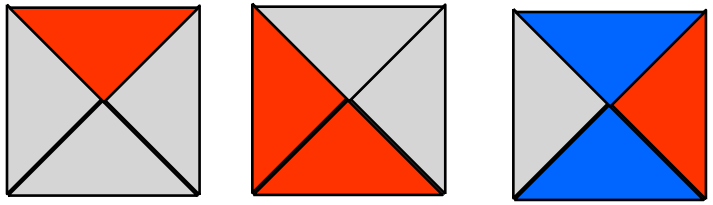


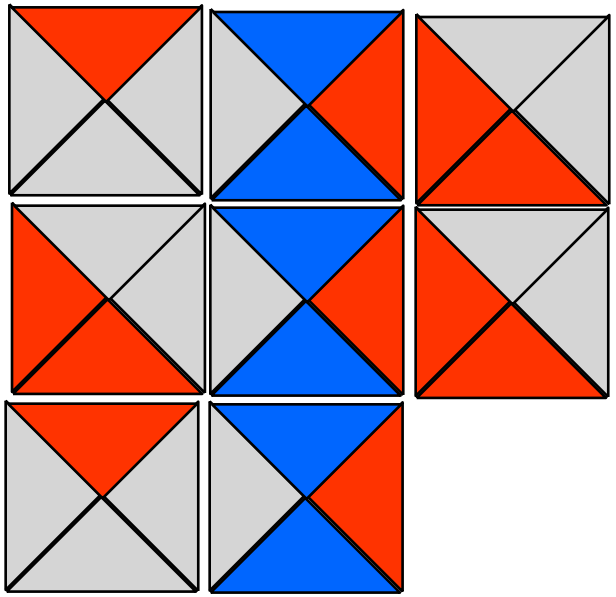


Tiling problem: lõpliku pinna katmine

- **Can I use this set of tiles to tile the floor of a house?**
  - **Lahenduv:** proovime järele kõikvõimalikud paigutused
  - Kuna maja põrand on lõplik, siis proovitavaid variante on lõplik (kuigi väga suur) hulk.
  - Võib mh meelde jätta, et see on NP-täieliku keerukusega ülesanne.
- **Selgub, et mõne plaatide hulgaga saab, mõnega ei!**
  - Lihtne on näiteks juhtum, kus on ainult ühte tüüpi plaate: valge.
  - Aga vaatame järgmist näidet (kolm erinevat plaaditüüpi) , kus ruudukujulise pinna plaatimine ei õnnestu.

Another example tile set:





Error

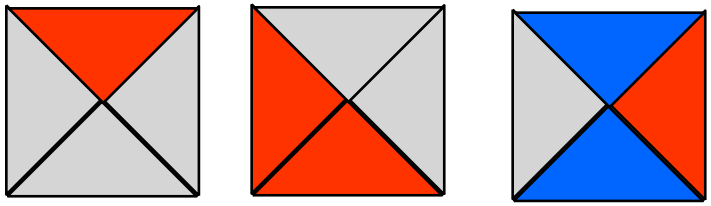
No Tiling for even small areas

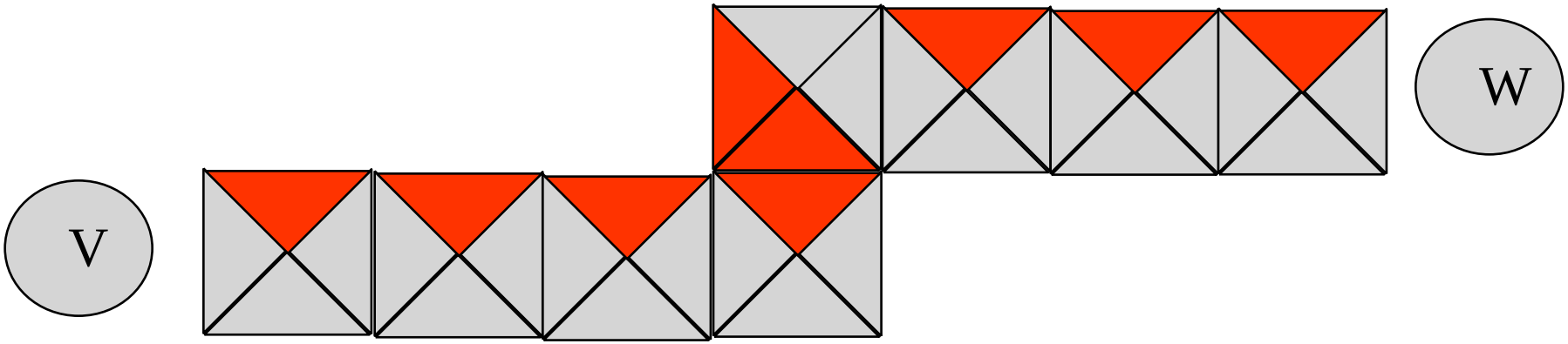
Tiling problem: undecidable question

- **Can a room of any size be tiled using this set of tiles?**
- **St, pind suurusega 2*2, pind 3*3, pind 4*4, jne jne ....**
  - **Ei ole lahenduv!**
  - Pane tähele, et **järeleproovimise meetod** enam ei anna kindlaid tulemusi, sest järgi tuleks proovida lõpmatult palju pinnasuurusi.
  - Pane tähele ka seda, et kui vastus on negatiivne (mingi suurusega N*N pinda ei saa meie plaatidega katta), siis mehaaniline järeleproovimine lõpuks selle avastab.
  - Raskus tekib, kui tegelikult saab kõiki pindu katta: järeleproovimise meetod proovib üha suuremaid ja suuremaid pindu, lõpmatuseni.

Analoogiline näide: “Domino Snakes”

- **Is it possible to connect V to W using a tile snake?**





“Domino Snakes” kohta saab tõestada, et:

Lets look at three cases:

- If the plane to lay down the snake in is finite?
  - **trivially decidable**
- If snakes can go anywhere in the plane?
  - **decidable**
- If snakes can only go in half of the plane?
  - **undecidable**

Kuulus näide matemaatikast

- In 1900, mathematician David Hilbert identified 23 mathematical problems and posed them as a challenge for the coming century.
- **The tenth problem asks for an algorithm to test whether a polynomial has an integral root**
- Apparently, Hilbert assumed that such an algorithm must exist.
- **This problem is unsolvable** (Matijasevic 1970)

Poollahenduvus

- Olgu ülesandeks tuvastada, **kas täisarv X kuulub mingisse lõpmatusse täisarvude alamhulka H**.
  - Mõne H jaoks on ülesanne **lahenduv**: näiteks, kui H on paarisarvude hulk, kui H on algarvude hulk jne,
  - Mõne H jaoks ülesanne **ei ole lahenduv**: näiteks, kui H on arvude hulk, millele vastavad programmid peatuvad.

- **Poollahenduvus** tähendab, et kui X juhuslikult kuulub hulka H, siis me saame seda algoritmiga alati näidata. Kui ei kuulu H-i, siis ei saa alati.

- Peatumisprobleemi puhul: paneme X-le vastava programmi käima ja kui ta peatub, siis loomulikult teame, et ta kuulub hulka H
  - Kui ta aga ei peatu, siis meil ei ole kindlat viisi aru saada, et ta ei kuulu hulka H.
  - **Peatumisprobleem on poollahenduv**.

- **On olemas ülesandeid, mis ei ole ka mitte poollahenduvad.**

Lahenduvus: muud

- **Vanad “vist ekslikud” oletused:**
  - Mathematics is consistent. Roughly this means that we cannot prove a statement and its opposite; we cannot prove something horrible like 1=2.
  - Mathematics is complete. Roughly this means that every true mathematical assertion can be proven i.e. every mathematical assertion can either be proven or disproven.
  - Mathematics is decidable. This means that for every type of mathematical problem there is an algorithm that, in theory at least, can be mechanically followed to give a solution. We say “in theory” because following the algorithm might take a million years and still be finite

Lahenduvus: muud

- **Hiljem selgus, et:**

- **In 1930, Kurt Godel shocked the world by proving that 1 and 2 cannot both be true**
  - Either you can prove false statements or there are true statements that are not provable.
  - Most people believe that mathematics is incomplete rather than mathematics is inconsistent
- **Turing was one of the people who showed that (3) is false by his work on Turing machines and the halting problem**